knowledge

MDPI

Article

Enhancing Chatbot Performance in a SaaS Platform Through Retrieval-Augmented Generation and Prompt Engineering: A Case Study in Behavioral Safety Analysis

Jorge Rivera [1,†], Scarlett Zapata [1,†], Ricardo Pizarro [2] and Brian Keith [1,*]

[1] Department of Computing & Systems Engineering, Universidad Católica del Norte, Antofagasta 1270709, Chile; jorge.rivera01@alumnos.ucn.cl (J.R.); scarlett.zapata@alumnos.ucn.cl (S.Z.)
[2] Pignus, Antofagasta 1271452, Chile
* Correspondence: brian.keith@ucn.cl
† These authors contributed equally to this work.

Abstract

This article presents a case study showing the development of a chatbot, named Selene, in a Software-as-a-Service platform for behavioral analysis using Retrieval-Augmented Generation (RAG) integrating domain-specific knowledge and enforcing adherence to organizational rules to improve response quality. Selene is designed to provide deep analyses and practical recommendations that help users optimize organizational behavioral development. To ensure that the RAG pipeline had updated information, we implemented an Extract, Transform, and Load process that updated the knowledge base of the pipeline daily and applied prompt engineering to ensure compliance with organizational rules and directives, using GPT-4 as the underlying language model of the chatbot, which was the state-of-the-art model at the time of deployment. We followed the Generative AI Project Life Cycle Frameworkas the basic methodology to develop this system. To evaluate Selene, we used the DeepEval library, showing that it provides appropriate responses and aligning with organizational rules. Our results show that the system achieves high answer relevancy in 78% of the test cases achieved and a complete absence of bias and toxicity issues. This work provides practical insights for organizations deploying similar knowledge-based chatbot systems.

Keywords: information retrieval; prompt engineering; large language models; behavioral analysis; software-as-a-service

Academic Editor: Sharifu Ura

Received: 19 August 2025
Revised: 20 October 2025
Accepted: 27 October 2025
Published: 5 November 2025

Citation: Rivera, J.; Zapata, S.; Pizarro, R.; Keith, B. Enhancing Chatbot Performance in a SaaS Platform Through Retrieval-Augmented Generation and Prompt Engineering: A Case Study in Behavioral Safety Analysis. *Knowledge* **2025**, *5*, 25. https://doi.org/10.3390/knowledge5040025

1. Introduction

Chatbots began as simple rule-based systems, but they have since become complex virtual assistants, customer service agents, and knowledge-sharing platforms that serve millions of users [1]. Recent developments in artificial intelligence (AI) and natural language processing (NLP) have allowed these systems to conduct conversations that resemble human interaction [2]. In particular, the emergence of Large Language Models (LLMs) has accelerated the development of chatbots, enabling them to better understand their interactions, generate relevant responses given a specific context, and adapt their behavior based on specific user needs through human–computer interaction [3].

In this work, we present a case study on the development of a chatbot in the behavioral analysis domain [4–6]. In particular, behavior analysis studies human behavior in organizational contexts to understand, predict, and influence it [7,8]. In particular, practitioners

in this field analyze behaviors along with their antecedents and consequences and then design interventions to promote desired outcomes [9].

Building chatbots for organizational use presents two primary obstacles. First, language models must use updated organizational data while correctly interpreting domain-specific terminology that general-purpose systems frequently misunderstand [10]. Second, organizations operate under strict regulatory frameworks that impose additional compliance requirements (e.g., finance, healthcare, and human resources information).

In this context, Retrieval-Augmented Generation (RAG) [11] and prompt engineering [12] can help address these obstacles. RAG connects model outputs to authoritative source documents. Prompt engineering shapes how queries are constructed and how responses are formatted, which improves both accuracy and completeness [13].

This work presents our implementation of RAG and prompt engineering in Selene, a chatbot originally developed using GPT-4 [14]. The Selene chatbot delivers behavioral analyses and actionable recommendations to support organizational development initiatives. To keep the chatbot updated with the latest information, we use an automated Extract, Transform, and Load (ETL) process [15] that refreshes the knowledge base daily. We present an overview of the system in Figure 1.

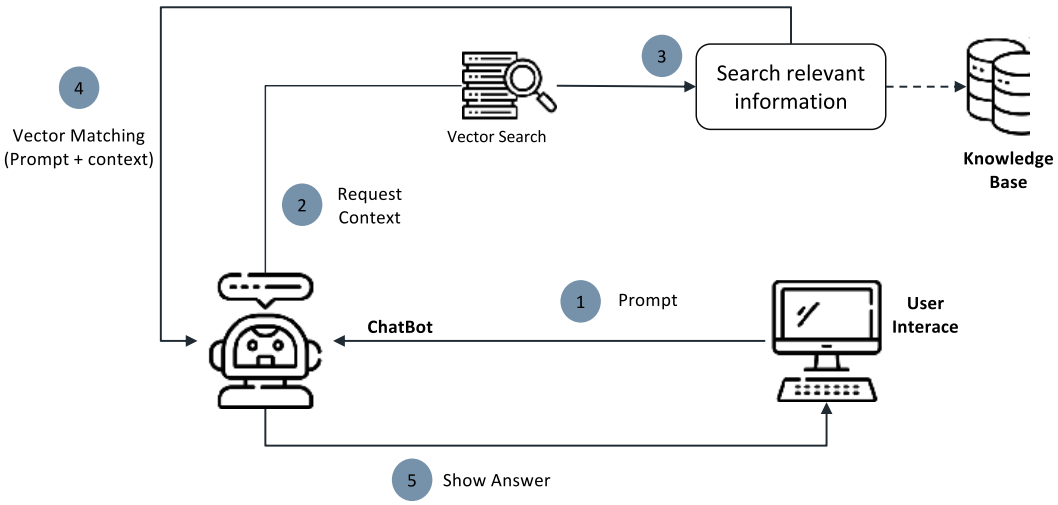

Figure 1. Overview of the Selene Chatbot within a SaaS platform for behavioral analysis. First, users submit queries through the web interface, which forwards requests to the chatbot microservice. GPT-4 with function calling then determines what information is needed to answer the query. The system retrieves relevant documents with a vector similarity search in the knowledge base. The system combines this retrieved context with the original query to create the final prompt, and GPT-4 generates a response grounded in the retrieved information.

We were motivated by three main challenges in this multi-tenant SaaS deployment. The chatbot lacked an automated mechanism to update its knowledge base, resulting in responses based on outdated information. It also struggled with domain-specific terminology and concepts unique to behavioral safety analysis, limiting its ability to provide accurate and professionally appropriate responses. Furthermore, the final chatbot version required proper role-based access control (RBAC) to ensure that the RAG pipeline did not retrieve forbidden information.

Thus, this paper contributes to the literature by presenting a case study on the integration of RAG and prompt engineering in a production behavioral safety platform. Rather than a controlled research experiment, we document the practical challenges and design decisions involved in deploying conversational AI in a specialized enterprise domain. Our contributions are twofold: **(1)** we demonstrate how automated knowledge maintenance combined with RAG can serve domain-specific information and **(2)** we provide evaluation

baselines and practical insights for practitioners facing similar deployment contexts. We focus on documenting real-world implementation decisions, identified failure modes, and lessons learned rather than comparing architectural alternatives, which, while valuable for future research, exceeded the scope of this organizational deployment.

Section 2 gives a summary of work related to chatbot development, RAG, and prompt engineering. Section 3 gives a summary of our methodology. Section 4 provides the details of the ETL process, prompt engineering patterns, and the RAG pipeline. Section 5 shows the results of our evaluation. Section 6 discusses the findings and lessons learned. Finally, Section 7 closes by presenting the conclusions and exploring possible future research directions.

2. Background and Related Work

2.1. Chatbot Development

Chatbots have attracted interest from both academia and industry in the past few years due to the advent of LLMs improving their interaction capabilities [1,16]. Historically, ELIZA [17] and ALICE [18] were among the first chatbot systems that utilized simple pattern matching and rule-based methods to simulate conversations. However, these methods faced issues with scalability and applicability across many domains [2].

In contrast, recent years have seen the rise of deep learning and neural networks, which has revolutionized the construction of chatbots. Currently, systems can utilize LLMs such as GPT [19], BERT [20], and their derivatives [21]. Following training on large datasets of text, these models are capable of generating coherent and contextually relevant responses [22]. Researchers have evaluated different architectures and methodologies to enhance chatbot performance, including attention mechanisms [23], memory networks [24], and reinforcement learning [25].

2.2. Retrieval-Augmented Generation

RAG systems combine retrieval-based and generation-based methodologies for NLP tasks [11]. In particular, these systems have a component that retrieves relevant elements from an outside knowledge source—the knowledge base—and a generative model that uses the aforementioned elements as additional contextual information to create the final output [26].

RAG has been employed in several tasks, including question answering [11], dialogue generation [27], and summarization [28]. In these tasks, RAG models provide more precise responses than models that rely solely on generation-based techniques, as they incorporate external knowledge [11]. The effectiveness of RAG is influenced by the quality and relevance of the retrieved data.

Table 1 shows some use cases of RAG and compares them with our case study. Recent advancements have extended RAG beyond research-oriented systems to specialized enterprise domains defined by stringent accuracy and compliance standards. Hu et al. [29] studied the risks of LLM-generated content and showed that false information can be categorized as more relevant, making it harder to keep information accurate. Hang et al. [30] developed a graph-based RAG utilizing semantic knowledge graphs that are continually updated to verify medical claims.

Table 1. Comparison of RAG implementations.

Approach	Retrieval Method	Knowledge Integration	Domain Specificity	Production Status
Lewis et al. [11]	Dense retrieval	Neural retriever with Wikipedia index	General-purpose QA	Research prototype
DPR [31]	Dense dual-encoder	BERT-based passage retrieval	Open-domain QA	Research, widely adopted
REALM [32]	Dense with pre-training	Joint retriever–encoder pre-training	General NLP tasks	Research prototype
MedGraphRAG [33]	Graph-based method	Medical knowledge graphs	Healthcare domain	Research prototype
This work	Dense with HNSW	Vector search with dynamic ETL synchronization	Behavioral safety	Production deployment

While these examples demonstrate the effectiveness of RAG in general contexts, our case study in behavioral safety analysis in an SaaS environment has domain-specific requirements beyond those found in general-purpose question-answering systems and enterprise applications. First, temporal evaluation tracking must distinguish between current worker status and historical worker information while maintaining context across multi-turn interactions. Second, the system must understand how to navigate the hierarchical structures (e.g., how departmental structures and supervisory relationships affect the semantic interpretation of queries). In this context, our platform implements RBAC through backend authentication before queries reach the RAG pipeline. Third, the interpretation of numerical thresholds must categorize continuous safety probability values into meaningful intervention levels with appropriate recommendations.

2.3. Prompt Engineering

Prompt engineering is the process of designing and refining input prompts to steer language models toward the desired outputs [12]. This process has become standard to use with pre-trained LLMs such as GPT-3 [34] and its successors, which can be adapted to different tasks using prompt engineering [13].

Researchers have examined various prompt engineering methodologies, including manual prompt design [35], automated prompt search [36], and continuous prompt optimization [37]. Moreover, empirical evidence shows that prompt design has a significant effect on the quality and relevance of the outputs of the models. Prompt design includes formatting [38] and the integration of domain knowledge and examples [39].

Moreover, we note that in our particular case, behavioral safety domains require interpreting numerical thresholds to handle the safety probability levels of the workers, where values are categorized into Low, Medium-Low, Medium-High, and High ranges with specific implications for organizational intervention strategies. In addition, the management of temporal context must maintain awareness of evaluation periods to keep track of historical worker information. Also, the navigation of the hierarchical organizational structure requires prompts that enable the model to interpret organizational hierarchy and roles correctly when formulating responses. Finally, the backend system enforces access boundaries through authentication and authorization mechanisms.

2.4. Positioning This Work

Our work contributes a specialized case study in the behavior analysis domain. Our novelty lies not in inventing new RAG mechanisms but in demonstrating their practical integration in this specific case. Behavioral safety analysis in our specific case study introduces requirements not addressed in general-purpose RAG implementations: strict role-based access control preventing users from accessing information across organizational boundaries, temporal evaluation tracking spanning months or years while keeping up with updated data requirements, and multi-level organizational hierarchy navigation where access permissions vary by role. We document how RAG and prompt engineering address these requirements in production, providing insights for practitioners deploying similar systems.

3. Methodology

Our approach takes elements from the Generative AI Project Life Cycle Framework described by Fregly et al. [40] into an iterative incremental software development process. In particular, the process included the following phases: data preparation, model experimentation, prompt engineering, evaluation, and deployment.

3.1. Requirements Engineering and Analysis

We started with the requirements engineering process to define the scope and requirements of the new chatbot version [41]. First, we identified the three key stakeholders that were required to define the needs of the system: the CEO, the CTO, and an expert in behavioral analysis. Through collaborative meetings, we were able to gather needs and learn about the issues with the pre-existing implementation of the chatbot that did not employ RAG or the right prompt engineering through.

Requirements engineering revealed three main challenges: First, the chatbot lacked an automated system to update or incorporate new data into its model, resulting in outdated information being provided to the user. Furthermore, the system could not handle queries involving domain-specific terms or organizational processes and it lacked proper data isolation, allowing access to data across client organizations, which needed to be corrected before properly using RAG.

3.2. Infrastructure Analysis

We examined the current architecture of the system and its underlying GPT-4 model to understand its capabilities and limitations. Specifically, the system operated through a microservices architecture, which allowed the interaction between the chatbot component and other components such as production databases, external AI services, and the web frontend.

In the basic implementation, the chatbot consumed information from CSV files containing all evaluations performed by client organizations (i.e., a preliminary form of RAG). To implement proper RAG, we explored the options provided by Microsoft Azure solutions, as Pignus had access to this infrastructure. In particular, we ended up selecting Azure Cosmos DB, which provides capabilities for storing and managing vectors essential for representing documents mathematically using embeddings [42].

3.3. Proposed Solution

Our solution considers three key components: ETL, prompt engineering, and RAG. Figure 2 presents the proposed system's architecture. The ETL pipeline extracts worker evaluations and relevant documentation from the MySQL database. Then, it transforms the content through normalization and embedding generation. Then, the content is loaded

into Azure Cosmos DB with vector indexing, allowing it to be used in conversations. User queries flow from the user interface through the Flask API to GPT-4, which then uses function calling to retrieve relevant context through vector similarity search. The generated responses are logged and returned to users through the web interface.

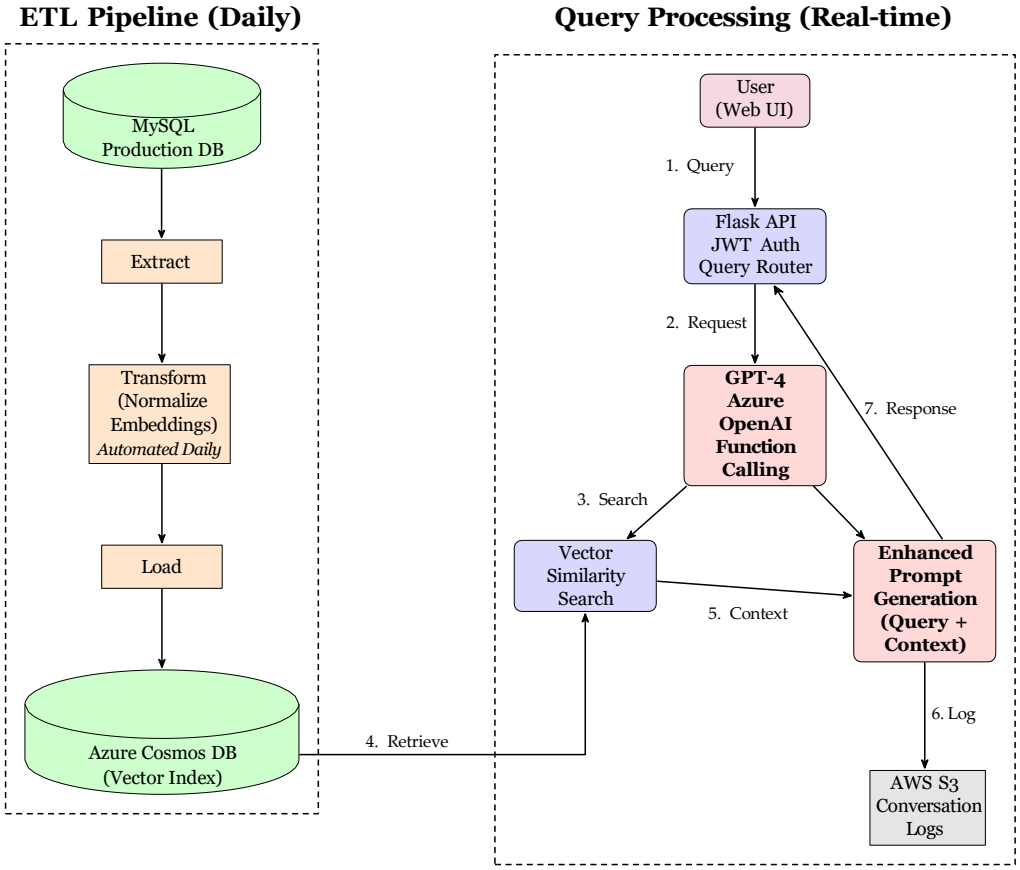

Figure 2. System architecture of the Selene chatbot.

3.3.1. Component 1: Automated ETL Process

The first component addresses the use of obsolete information by the Selene chatbot through an ETL process that runs daily. In particular, during the extraction phase, we use the MySQL database to retrieve worker evaluation records, organizational structures, and other documents that have been updated or created since the last update. After this initial phase, the transformation phase normalizes the text (e.g., by eliminating accents and converting to lowercase) while keeping domain terminology via a protected word list. During this phase, we also generate embedding representations using the `text-embedding-ada-002` model from OpenAI (San Francisco, CA, USA). We note that the embeddings are then used for similarity search in the RAG component. Finally, the loading phase updates Azure Cosmos DB in an incremental fashion, inserting new records created since the last update and updating any modified entries.

3.3.2. Component 2: Prompt Engineering Patterns

The second component uses prompt engineering methods to ensure that the behavior of the chatbot is appropriate based on the requirements analysis. In particular, we applied several design patterns based on the work of Schmidt et al. [43]. First, the persona pattern configures the model so that it takes the role of a professional behavioral safety assistant. Next, the template pattern provides the model with structured frameworks to respond to different query types, based on predefined types obtained from the requirements engineer-

ing process. Furthermore, the context manager pattern maintains conversation coherence across multiple exchanges. Finally, the input semantics pattern defines guidelines to help handle numerical thresholds and temporal references.

We show general examples of prompt-engineering techniques in Figure 3. Several prompt design patterns were applied to improve the interaction with the model [43], including the persona pattern, template pattern, context manager, and input semantics patterns. These patterns help establish the chatbot's role, provide structured frameworks for queries, maintain conversation coherence, and define clear guidelines for data input.

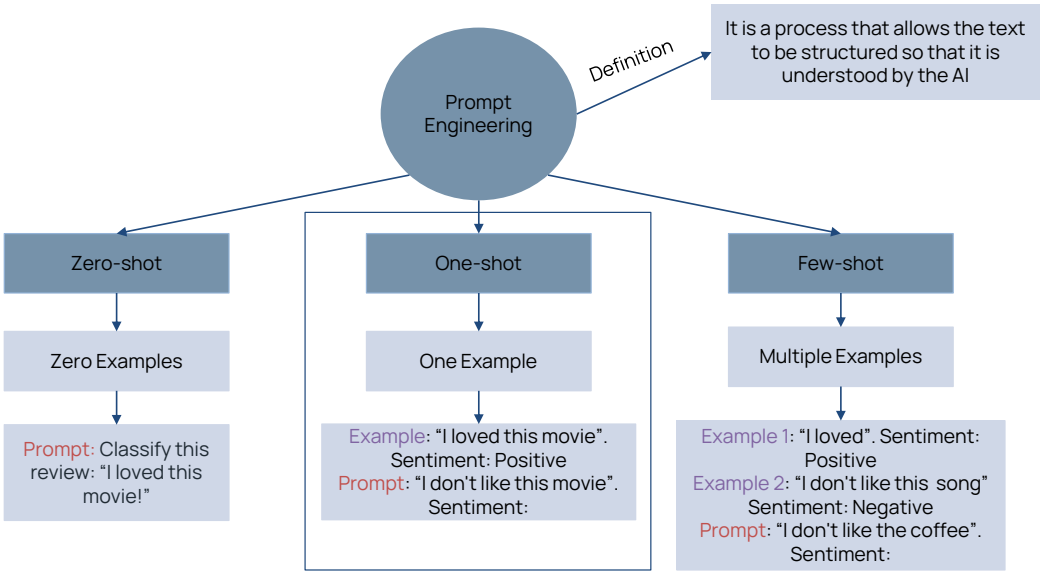

Figure 3. Prompt engineering examples for zero-shot, one-shot, and few-shot prompting.

3.3.3. Component 3: Retrieval-Augmented Generation

The third component employs RAG to enable the chatbot to retrieve relevant information from an external knowledge base and incorporate it into the generated responses. The retrieval mechanism uses vector similarity search with HNSW indexing and cosine distance metrics to identify relevant documents. The retrieved context includes both organizational evaluation data and platform documentation, filtered according to the user's organizational role and access permissions. The generation phase combines the information recovered with the prompt engineering patterns to produce accurate and contextually appropriate responses.

4. Implementation

4.1. ETL Process Implementation

The extraction phase considers multiple data sources, including a MySQL library, worker information, evaluation data for both supervisors and operators, and organizational hierarchical structures. We also extracted relevant information from the help section of the platform itself, which contained HTML files, videos, and PDFs. We converted all HTML files and video transcripts to plain text, while processing PDF files using the PyMuPDF library.

The automated ETL process directly addresses the stale information challenge identified during requirements engineering. We run the process daily during off-peak hours. During the extraction process, the system queries the MySQL production database for evaluation records modified since the last ETL execution, checks the platform help documentation directory for new or modified files based on file modification times, and retrieves any content that has been modified. The transformation phase generates fresh embeddings for the updated content, which avoids unnecessary API calls for unchanged data. The load-

ing phase performs incremental updates to Azure Cosmos DB collections, inserting new records and updating modified entries while preserving unchanged data. This incremental approach keeps the chatbot operating on current organizational data without full database rebuilds. The ETL execution history is logged to enable troubleshooting and to monitor data currency.

The transformation phase standardized the extracted data. First, all text data undergo normalization, including the removal of accents (e.g., 'información' becomes 'informacion', meaning 'information' in Spanish), conversion to lowercase for case-insensitive matching, and elimination of unnecessary spaces while preserving domain-specific terminology through a protected term list, which were part of a specialized terminology glossary developed collaboratively with behavioral safety experts during the requirements engineering phase of the project. Temporal data are converted to UTC format with timezone metadata preservation. The transformation process also includes semantic enrichment through embedding generation using OpenAI's `text-embedding-ada-002` model, which produces 1536-dimensional vectors. Furthermore, function calling schemas were designed for the three organizational user role levels (Administrator, Multi-Company, Single-Company), and prompt engineering patterns encoded behavioral safety evaluation frameworks and competency definitions.

The loading phase populates Azure Cosmos DB for MongoDB vCore with the transformed data. The database schema is designed to support efficient vector similarity search using Hierarchical Navigable Small World (HNSW) indexing with cosine similarity as the distance metric, enabling fast approximate nearest neighbor search with cosine similarity as the distance metric. Each document in the database includes the original text content, the embedding vector, metadata for filtering (e.g., organization ID and user access level), and temporal information for tracking data currency.

4.2. Prompt Engineering Patterns

Our prompt engineering implementation follows Schmidt et al.'s [43] categorization. Based on the general prompt engineering patterns shown in Figure 3, we develop multiple prompts to address a diverse range of user queries. Four main patterns were implemented to guide the behavior of the chatbot and ensure appropriate responses. We note that these patterns were validated by the organizational stakeholders as part of the development process. Table 2 shows example prompts.

Table 2. Example prompts for chatbot evaluation (summarized due to confidentiality restrictions).

Prompt ID	Prompt Text
P1	"Your name is Selene, you are a virtual assistant…"
P2	"The probability of safe working conditions is a decimal number between 0 and 1. The levels are: …"
P3	"The competencies for supervisory workers are: …"
P4	"The function …allows you to obtain information about the workers in the following way: …"
P5	"The function …allows you to consult the administration of …"
P6	"For the response of the function …, you must indicate where the information is located"

The persona pattern configures the Selene chatbot to assume the role of a virtual assistant and expert in behavioral safety analysis. Furthermore, we configure the chatbot to ensure that it uses a professional and friendly tone. This affects the style and content of the answers, ensuring that they are all consistent for all types of questions. The persona definition includes clear limits and boundaries for what information can be accessed and

what medical or legal advice can be given. For example, a persona cannot access data outside of their assigned organizational boundaries.

The template pattern provides structured frameworks for the chatbot to use in different query types. We defined different templates for statistical queries, individual worker queries, comparative analyzes, and procedural guidance. The system then automatically selects the appropriate templates based on the type of query from the predefined templates.

The context manager pattern helps the chatbot maintain conversations coherent and continuous by keeping track of previous dialogue exchanges and user preferences. In particular, this pattern manages context using a sliding window approach with retention based on relevance. Through this approach, only the most relevant information is kept within the token limit of the underlying LLM.

The input semantics pattern establishes rules and formats for input data, including formal specifications of domain entities such as workers, evaluations, and competencies. These patterns define metrics such as the probability of safe work, which ranges from 0 to 1 and is categorized into four levels: Low [0–0.25), Medium-Low [0.25–0.5), Medium-High [0.5–0.75), and High [0.75–1.0].

4.3. RAG Pipeline Implementation

We implemented the RAG pipeline to retrieve relevant information from the knowledge base for query responses. The pipeline operates in five steps, illustrated in Figure 4.

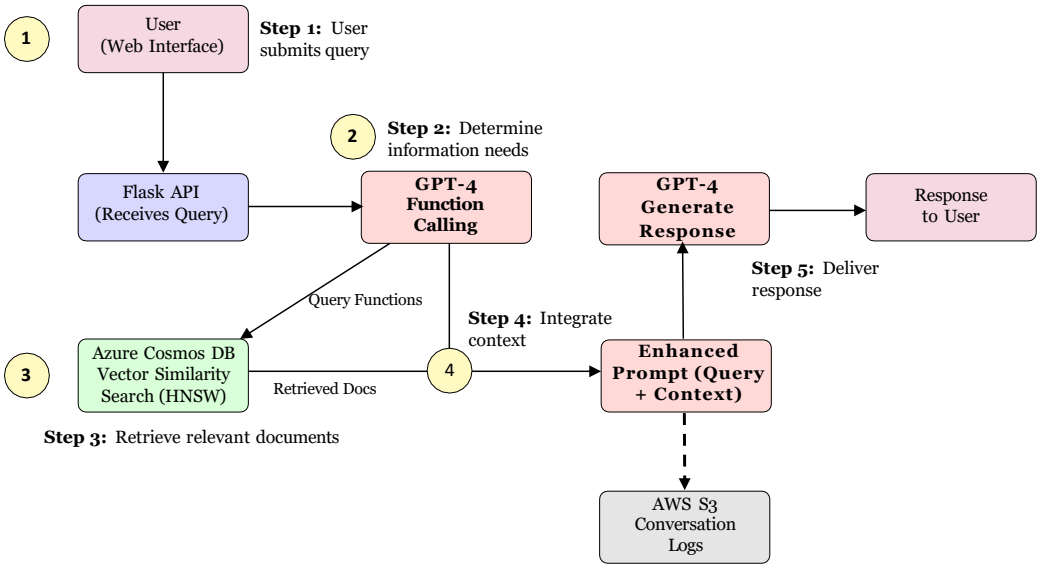

Figure 4. Five-step RAG pipeline. Users submit queries, GPT-4 function calling determines information needs, vector similarity search retrieves relevant documents from the knowledge base, context integrates with the prompt, and GPT-4 generates and delivers the response.

4.3.1. First Step: Query Reception

Once the system receives a query through the frontend, the Flask API microservice relays the request to the backend. The system validates the authentication token, extracts the user's basic information (identity and organizational role), and then logs the query for audit purposes. The underlying language model and the RAG pipeline (if required) can now proceed to handle the query.

4.3.2. Second Step: Function Calling

In particular, the system uses GPT-4's function calling feature to determine what information it needs to answer the question. More specifically, the model examines the

query and chooses whether to use the predefined functions to get the needed information from the production database and/or the knowledge base, letting the model take into account additional information in their context window. The function calling approach was configured through prompt engineering patterns that established response formats and calling methods. Furthermore, we note that the chatbot respects role-based permissions (Administrator, Multi-Company, and Single-Company users) with different access levels within the platform.

4.3.3. Third Step: Vector Similarity Search

The system retrieves relevant documents by using a vector similarity search in the knowledge base. The system generates an embedding vector for the query using the same `text-embedding-ada-002` model used during ETL transformation. This embedding queries Azure Cosmos DB with HNSW indexing. The search retrieves the most relevant documents based on cosine similarity. The system enforces role-based access control through backend authentication and authorization mechanisms before queries reach the retrieval pipeline. JSON Web Token validation extracts the user identity and organizational role, ensuring that users can only access information within their authorized scope. This architectural decision prevents cross-organizational information leakage at the system level instead of relying on retrieval-time filtering.

4.3.4. Fourth Step: Context Integration

The system now integrates the retrieved context with the original query to create an enhanced prompt. The system combines the user's question, relevant retrieved documents, function call results, and prompt engineering patterns into the final prompt for the generation model, which provides GPT-4 with all the necessary context.

4.3.5. Fifth Step: Response Generation

The final step uses GPT-4 to generate the response. The response is then delivered to the user via the Flask API to the web interface for display purposes. We note that the generated response is logged into AWS S3 to track conversation history tracking for audit purposes.

4.3.6. Failure Modes and Error Pattern Analysis

We analyze the behavior of the system to identify relevant failure patterns. We note that contextual precision failures occurred in 5 test cases (approximately 22%), where the system retrieved relevant background information but failed to surface specific data points needed for precise answers. For example, when queried about a worker's exact competency score, the system sometimes retrieved general documentation about the competency framework without the evaluation of the specific individual that was requested. Contextual relevancy failures occurred in seven cases (approximately 30%), which involved the retrieval of tangentially related but not directly applicable information. This pattern appeared most frequently with multi-dimensional queries that needed the navigation of hierarchical organizational structures. For example, queries about specific departments within subsidiary companies sometimes retrieved information about other departments or parent company data. Hallucinations occurred in two cases (approximately 9%), where the model generated factually incorrect information that was not supported by the retrieved context. In both instances, the queries requested specific numerical values that were not present in the retrieved documents. The vector similarity search retrieved documents containing related information, but did not include any document that allowed the system to respond to the query correctly. However, rather than acknowledging that it had

insufficient information, the model attempted to generate an answer anyway, resulting in hallucinations.

4.3.7. Architecture and Performance Characteristics

We note that in the current implementation of the RAG pipeline, each user query triggers a complete pipeline execution. Thus, additional engineering work for production scaling, such as latency improvements through caching strategies, API orchestration optimization with parallel function calls, and connection pooling represent, remain beyond the scope of this case study.

5. Results and Evaluation

5.1. Evaluation Methodology

We evaluated the enhanced Selene chatbot using automated DeepEval metrics [44] followed up by organizational qualitative validation. The evaluation assessed both technical performance of the RAG pipeline and the perception of key stakeholders from Pignus.

5.1.1. Evaluation Tools

We considered LangChain [45], MLflow [46], and DeepEval [44] as evaluation alternatives. The first approach that we considered was LangChain [45] for managing language model applications, but we rejected it due to costs and mandatory dependencies that were not aligned with the project requirements. We then considered MLflow [46], which is a library focused on machine learning experiment management. However, it lacked the specific and functional metrics required for chatbot evaluation. We ultimately selected DeepEval [44], an open-source evaluation framework designed for language models with built-in RAG support, due to its range of metrics designed for language models, including precision, coherence, hallucination, and relevance.

5.1.2. Evaluation Process

To test our model, we designed two sets of queries along with their expected output. The first set of queries was designed to assess the quality of the RAG pipeline, while the second set of queries was designed to evaluate the general response quality. The complete evaluation set comprises 23 test cases based on frequently asked questions that were identified during the requirements engineering process with the help of organizational stakeholders. We note that the platform processes an average of about 230 individual worker evaluation reports monthly based on 2023 operational data, providing the underlying data pool for chatbot queries. The knowledge base includes worker evaluation records that span multiple months, organizational hierarchical data that define company structures and access relationships, and platform documentation that covers system usage and behavioral safety concepts.

Each test case includes five components: an input query representing actual user information needs, an expected output validated by domain experts from the organization defining the correct response, an actual output generated by the enhanced chatbot, retrieval context showing what documents were retrieved from the knowledge base, and conversation context capturing relevant prior exchanges in multi-turn interactions. Thus, the fields employed for the use cases were as follows:

- `input`: This element corresponds to the question asked by the user.
- `expected_output`: This element corresponds to the expected answer.
- `actual_output`: This element corresponds to the actual answer generated by the chatbot.
- `retrieval_context`: This element corresponds to the context of the RAG.
- `context`: This element corresponds to the additional data received by the LLM.

To obtain the input and expected_output fields, the datasets were iterated over. The actual_output was generated by invoking the model to produce a response. The retrieval_context was acquired from the files stored in the database used to apply RAG, while the context was set according to the additional data. We show examples of user queries used throughout the evaluation process in Table 3. We note that we had a total of 23 different test cases in the evaluation.

Table 3. Example evaluation queries. Queries modified to preserve company confidentiality.

Test Cases	Example Expected Response
T1: Help with area management	"To manage the company's areas, use the sidebar and follow these steps: . . . "
T3: Safe work analysis	"There are . . . workers who have a safe work probability at level . . . "
T4: Competence analysis	"There are . . . workers who have competence . . . at level . . . "
T5: Safest worker evaluation	"The worker with the highest safe work probability is . . . "
T6: Most developed competence	"The most developed competence for workers is . . . "
T7: Specific worker query	"The information of the worker with ID number . . . is as follows: . . . "
T8: Worker evaluation history	"The evaluation history of the worker . . . is: . . . "

5.1.3. Evaluation Metrics

DeepEval provides eight evaluation metrics that assess different aspects of chatbot performance. The metrics are evaluated on a scale from 0 to 1, with higher scores indicating better performance. The results are categorized according to the organizational standards used for the evaluation of behavioral competence across the platform: Low [0–0.25), Medium-Low [0.25–0.5), Medium-High [0.5–0.75) and High [0.75–1.0]. We note that these categories are aligned with the usual terminology of the organizational stakeholders.

We first describe the general evaluation metrics that assess response quality without considering the RAG pipeline and additional context:

- Answer Relevancy: Measures the quality and relevance of the response to the question, ensuring relevant and useful answers [42,47].
- Bias: Detects biases in the response (e.g., gender, racial, or political), ensuring impartial and fair answers [48,49].
- Toxicity: Identifies toxic elements in the response (e.g., mockery, hatred, disdainful statements, or threats), maintaining safe and respectful interactions [49].

Next, we consider metrics that evaluate RAG pipeline quality. Some resemble traditional information retrieval metrics, such as precision and recall.

- Faithfulness: Evaluate whether the generated responses remain faithful to the retrieval context, ensuring accurate and reliable information [42,50].
- Contextual Precision: Measures the contextual precision of each element in the retrieval context based on the expected response, ensuring the chatbot uses the most relevant context information [42].
- Contextual Recall: Evaluates the model's ability to retrieve information aligned with the expected response and retrieval context, preventing omission of important information [42].
- Contextual Relevancy: Measures the relevance of the information in the retrieval context based on the question, ensuring coherent responses [42].

- Hallucination: Detects factually incorrect information or fabricated information by comparing it with the actual response and the context provided, preventing misleading responses [51].

After defining the evaluation metrics, we executed the tests and stored the results in two CSV files for analysis.

5.2. Evaluation Results

Table 4 presents the Selene chatbot. Figure 5 visualizes evaluation metrics across all 23 test cases. Answer relevancy and faithfulness show consistently high scores across most test cases, while contextual precision shows more variation, indicating opportunities for retrieval optimization. Thus, our RAG implementation showed promising results in enhancing the chatbot's ability to provide accurate and context-specific responses.

Table 4. Results of the RAG pipeline according to different metrics.

Metrics	Low	Medium-Low	Medium-High	High
Answer Relevancy	0	3	2	18
Faithfulness	1	0	0	22
Contextual Precision	5	10	2	6
Contextual Recall	6	0	1	16
Contextual Relevancy	7	0	0	16
Hallucination	2	0	0	21
Bias	0	0	0	23
Toxicity	0	0	0	23

With respect to the general evaluation metrics, the chatbot achieved perfect scores (23/23 High) in terms of bias and toxicity issues, showing no problems in both categories. Thus, the prompt engineering patterns that were applied helped the chatbot maintained a professional and neutral tone across all responses. Furthermore, faithfulness scores were also high, with 22 of 23 test cases (96%) achieving High scores. The only Low score (T15) occurred when the model generated information that was not fully supported by the retrieved context. Answer relevancy presented slightly more mixed results, with 18 of 23 cases (78%) achieving High scores, while 3 cases scored Medium-Low and 2 scored Medium-High. This shows that the model generally provided relevant responses, although occasional deviations occurred.

We now present the results of the RAG evaluation metrics. First, contextual precision presents the most challenges, as only 6 of 23 cases (26%) achieving High scores. The distribution shows 5 Low, 10 Medium-Low, and 2 Medium-High scores, which shows that the retrieval mechanism frequently retrieves relevant information but does not always rank the information optimally. Contextual recall performs better, with 16 of 23 cases (70%) achieving High scores, although 6 cases scored Low. These results suggest that the system can generally retrieves the necessary information, but will occasionally miss relevant documents. Contextual relevancy scored high in 16 test cases (70%) but failed in 7 instances (30%), indicating that retrieval sometimes identified tangentially related rather than directly applicable information. In the T20 and T21 cases, the contextual precision was lower than in the other cases (i.e., the retrieval context lacked the specific information required to answer). Comparing expected and actual responses for these cases revealed that the language model tends to generate more extensive responses. Furthermore, in the case of T20, this led the system to attempt to generate answers that hallucinated the missing data.

Only two test cases (T17 and T20, approximately 9%) presented hallucination issues. In these cases, the hallucinations occurred because the chatbot could not find the required

information to respond directly from the given context or failed to provide the requested information despite correctly identifying the worker and the company in internal functions. Rather than acknowledging insufficient information, the model attempted to generate responses anyway, resulting in factually incorrect output.

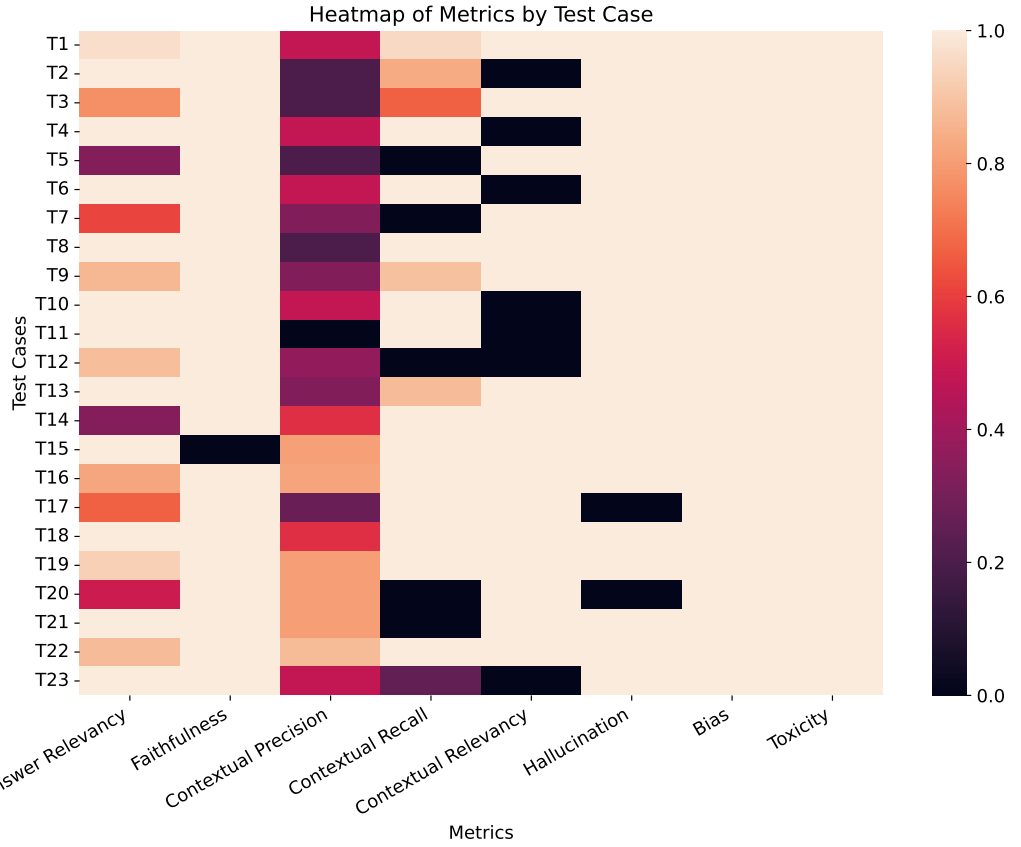

Figure 5. Heatmap visualization of evaluation metrics across all 23 test cases. Color scale: dark purple (0.0–0.2) represents Low performance, purple-pink (0.2–0.4) Medium-Low, orange (0.4–0.6) Medium-High, light orange (0.6–0.8) High, and white (0.8–1.0) Very High performance. Black cells indicate non-applicable metrics. Rows show individual test cases (T1–T23); columns show the eight evaluation metrics.

5.3. Organizational Validation

In addition to the quantitative measures from the preceding section, we validated that the enhanced chatbot met business requirements with the aforementioned key organizational stakeholders. In particular, Pignus' key stakeholders (CEO, CTO, and Behavioral Safety Specialist) participated throughout development, providing domain expertise and validating the system. All three stakeholders validated the chatbot's improvements, noting substantial gains in domain knowledge accuracy and response relevance compared to the baseline system.

Furthermore, multi-turn conversation sequences showed the effectiveness of the context manager pattern. For example, consider a representative interaction (constructed from typical usage patterns): A user could ask "How many workers have high safe work probability?" and they would get a reply like, "In Company A, 47 people have a high safe work probability (0.75–1.0). Do you want more information about a specific area or department?" Then, the user could ask, "Who is the least safe worker?" and the system would have to understand that the user is asking for a person in the previously mentioned company (i.e., keeping the context). Thus, the system should provide a response such as "The worker with the lowest safe work probability in Company A is Worker_X with a score

of 0.78. While still in the High category, this worker shows opportunities for improvement in analytical capacity and communication effectiveness."

We note that this validation method has some issues. First, these stakeholders were directly involved in the development process and had a vested interest in the system's success. Furthermore, this validation occurred during the pilot deployment with limited real-world usage. Thus, instead of assessing user adoption or satisfaction, we focused on showing technical feasibility. Subsequent studies would need to include assessments conducted by independent experts using blinded comparison techniques. However, such an evaluation was considered to be outside of the scope of this case study.

6. Discussion

Our results have shown that the system successfully provides accurate and contextually appropriate responses based in organizational data. We managed to eliminate bias and toxicity through prompt engineering patterns, effectively maintaining professional standards in the responses generated by the chatbot. However, the system showed issues with contextual precision. Thus, while the retrieval pipeline generally finds relevant information, it does not always retrieve the best documents for answering queries, particularly in queries requiring specific numerical values or detailed evaluations, where the system sometimes retrieved general background information rather than exact data points.

6.1. Deployment Characteristics and Scalability

The current production deployment costs approximately USD 70 per month, with most allocated to Azure Cosmos DB storage and compute resources. The smaller portions cover the usage of the Azure OpenAI API for embedding generation and completion requests, and AWS S3 storage for conversation logging. This aggregate cost reflects the currently limited scale of the first production pilot with limited concurrent users. We did not instrument per-query cost attribution in the current implementation. We also did not perform a detailed analysis of the memory consumption profile or latency during this case study. Performance profiling with percentile-based latency distributions, memory usage characterization under varying loads, and scalability testing represent important future engineering work for production-grade deployment.

6.2. Lessons Learned and Recommendations

In terms of lessons that may benefit practitioners working on similar projects, we have identified three key lessons. First, the requirements engineering process proved essential to the project. We spent three weeks in the process of understanding stakeholder needs and documenting domain-specific terminology. Identifying frequently asked questions during the requirements elicitation process informed the design of test cases, ensuring that evaluation focused on realistic cases rather than purely synthetic scenarios. Second, the selection of appropriate evaluation metrics is necessary to properly understand system performance. In this work, we used specialized RAG metrics from the DeepEval library. However, future work should supplement these metrics with human evaluation approaches that include blinded assessment. Third, our iterative development methodology enabled us to improve our solution based on the feedback from the key stakeholders. In particular, we were able to identify terminology and formatting issues with the generated responses early in the development phase since we had regular feedback meetings with the relevant stakeholders.

6.3. Limitations

While our case study results are promising, there are several issues that need to be examined as limitations.

Evaluation Scale and Statistical Power. First, our 23 test cases have a limited statistical power for generalization. These test cases were derived from frequently asked questions identified during the requirements engineering process. However, while these examples serve as useful as representative cases to test the RAG system, this approach does not have the sufficient scale and coverage to obtain statistical conclusions about the performance of the RAG system.

Language and Domain Constraints. The system primarily supports Spanish, hence limiting its applicability in contexts that require additional languages. However, the underlying embedding model of the RAG pipeline is multilingual (`text-embedding-ada-002`) and could support other common languages (e.g., English). However, we must note that the prompt engineering and terminology handling would require substantial modifications to be applicable in other languages or domains. Furthermore, we conducted the evaluation entirely within a single organizational and cultural context, raising questions about generalization to different industries, organizational structures, or regulatory environments.

Baseline Comparison Absence. We do not perform comparisons against baseline architectures such as LLM-only systems without RAG (beyond the key stakeholders' perception of improvement), traditional BM25 sparse retrieval methods, or hybrid approaches combining BM25 with neural rerankers in our experiments. Such studies represent important future work for validating RAG's value proposition in this domain but exceeded the scope of this organizational deployment case study.

Ablation Study Absence. We did not conduct ablation studies examining the impact of individual design choices, including parameters such as chunk size or top-k, adding query rewriting strategies, or using different embedding models, vector databases or indexing algorithms. Performing these ablation studies would allow us to identify which specific components drive performance and which represent over-engineering. However, we considered these to be outside of the scope of the current case study. Thus, ablation studies represent important future research directions for optimizing RAG systems in specialized domains.

Performance Profiling Gaps. We did not analyze latency profiles with percentile distributions, memory consumption characterization under varying loads, or cost attribution per query type. We also did not perform formal scalability testing with stress testing methodologies during this deployment. These performance metrics would be needed for scaling up production capacity. However, they were considered out of scope for the initial deployment. Performance engineering represents important future work for scaling beyond the pilot deployment.

7. Conclusions

We presented a case study about the Selene chatbot development process in a SaaS platform for behavioral safety analysis to improve its performance through RAG and prompt engineering. This work demonstrated the practical application of RAG and prompt engineering in a production environment. The lessons learned and identified patterns provide guidance for organizations that deploy domain-specific chatbots.

Our analysis of results shows that the improved chatbot achieves a high performance, with 78% high answer relevancy, 96% high faithfulness and a complete elimination of bias and toxicity. Furthermore, the prompt engineering patterns used in this study (persona, template, context manager, and input semantics) effectively helped the model generate consistent and professional responses. While some challenges remain, the system was able to provide valuable support for behavioral safety analysis tasks.

While our work shows the applicability of RAG in this domain, there are several limitations to our study that should be addressed by future work. First, the evaluation

should be expanded with extensive testing (e.g., 200–300 test cases generated through an automated question generation model using query synthesis techniques), which would allow us to report proper statistical information for the evaluation metrics. Once there exists a sufficient deployment history, it would be useful to analyze specific capabilities of the model by identifying the different types of queries logged by the system. Furthermore, implementing blind human evaluation protocols with independent expert panels would strengthen the methodology.

In addition, comparing with other baselines and performing ablation studies would help evaluate the design decisions made during this study. Comparisons against LLM-only systems without RAG, BM25 sparse retrieval, hybrid BM25 with neural reranking approaches, and alternative vector databases could help quantify the value of our RAG pipeline. Advanced retrieval techniques could be used to address the contextual precision challenges identified in the evaluation.

Moreover, extending language support beyond Spanish would broaden the applicability of the system. While the embedding model used in this work is multilingual, doing this would require language-specific prompt engineering adaptation to maintain quality across languages and cross-lingual retrieval evaluation methodologies. Furthermore, testing plans for each target language would need to address culturally appropriate terminology usage and response formatting conventions.

Performance engineering represents important work for scaling beyond pilot deployment. Implementing profiling with percentile-based latency tracking for the 50th, 90th, 95th, and 99th percentiles, cost attribution per query type, response caching for frequent queries, connection pooling for database and API services, and load balancing for concurrent user scaling would enable production-grade deployment. Formal stress tests would establish capacity limits and identify optimization priorities.

Advanced prompt engineering techniques could further enhance response quality. Chain-of-thought reasoning for complex multi-step queries, automated prompt optimization through reinforcement learning from human feedback, and self-consistency checking mechanisms to detect potential hallucinations before response delivery represent promising research directions.

Author Contributions: Conceptualization, R.P. and B.K.; methodology, J.R., S.Z. and B.K.; software, J.R. and S.Z.; validation, J.R., S.Z. and R.P.; formal analysis, J.R. and S.Z.; investigation, J.R. and S.Z.; resources, R.P.; data curation, J.R. and S.Z.; writing—original draft preparation, J.R. and S.Z.; writing—review and editing, J.R., S.Z., R.P. and B.K.; visualization, J.R. and S.Z.; supervision, B.K. and R.P.; project administration, B.K. and R.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by Universidad Católica del Norte through project 2023-11010033-VRIDT-UCN. Pignus received infrastructure support through the Microsoft for Startups Program. The APC was funded by Universidad Católica del Norte.

Institutional Review Board Statement: This study did not require ethical review as it involved technical evaluation of software performance using automated metrics (DeepEval framework). Organizational stakeholders participated as professional collaborators in software development, not as research participants.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data cannot be made publicly available due to organizational confidentiality agreements. Evaluation results are presented in full in the paper.

Acknowledgments: We would like to thank the Capstone Project Permanent Program at Universidad Católica del Norte for providing the opportunity to conduct this applied research. We are grateful to Pignus for their collaboration and support throughout this project, and we acknowledge the

Microsoft for Startups Program for their ongoing support of Pignus' technological and applied research initiatives. This work was partially supported by the project 2023-11010033-VRIDT-UCN.

Conflicts of Interest: Ricardo Pizarro is the CTO of Pignus. The other authors declare that they have no competing interests.

References

1. Følstad, A.; Brandtzæg, P.B. Chatbots and the new world of HCI. *Interactions* **2017**, *24*, 38–42. [CrossRef]
2. Dale, R. The return of the chatbots. *Nat. Lang. Eng.* **2016**, *22*, 811–817. [CrossRef]
3. Hadi, M.U.; Tashi, Q.A.; Qureshi, R.; Shah, A.; Muneer, A.; Irfan, M.; Zafar, A.; Shaikh, M.B.; Akhtar, N.; Hassan, S.Z.; et al. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Prepr.* **2025**. [CrossRef]
4. Chiu, Y.Y.; Sharma, A.; Lin, I.W.; Althoff, T. A Computational Framework for Behavioral Assessment of LLM Therapists. *arXiv* **2024**, arXiv:2401.00820. [CrossRef]
5. Kim, M.; Lee, H.; Park, J.; Lee, H.; Jung, K. AdvisorQA: Towards Helpful and Harmless Advice-seeking Question Answering with Collective Intelligence. *arXiv* **2024**, arXiv:2404.11826.
6. Aggarwal, A.; Tam, C.C.; Wu, D.; Li, X.; Qiao, S. Artificial intelligence–based chatbots for promoting health behavioral changes: Systematic review. *J. Med. Internet Res.* **2023**, *25*, e40789. [CrossRef]
7. Cooper, J.O.; Heron, T.E.; Heward, W.L. *Applied Behavior Analysis*; Pearson/Merrill-Prentice Hall: Upper Saddle River, NJ, USA, 2007; Volume 2.
8. Fisher, W.W.; Piazza, C.C.; Roane, H.S. *Handbook of Applied Behavior Analysis*; Guilford Publications: New York, NY, USA, 2021.
9. Mayer, G.R.; Sulzer-Azaroff, B.; Wallace, M. *Behavior Analysis for Lasting Change*; Sloan Pub.: Collinsville, IL, USA, 2012.
10. Chen, H.; Liu, X.; Yin, D.; Tang, J. A survey on dialogue systems: Recent advances and new frontiers. *ACM SIGKDD Explor. Newsl.* **2017**, *19*, 25–35. [CrossRef]
11. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.t.; Rocktäschel, T.; et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 9459–9474.
12. Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.* **2023**, *55*, 1–35. [CrossRef]
13. Gao, T.; Fisch, A.; Chen, D. Making Pre-trained Language Models Better Few-shot Learners. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Online, 1–6 August 2021; pp. 3816–3830.
14. Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F.L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. Gpt-4 technical report. *arXiv* **2023**, arXiv:2303.08774. [CrossRef]
15. Nwokeji, J.C.; Matovu, R. A systematic literature review on big data extraction, transformation and loading (etl). In *Intelligent Computing, Proceedings of the 2021 Computing Conference, Volume 2, Virtual, 15–16 July 2021*; Springer: Cham, Switzerland, 2021; pp. 308–324.
16. Dam, S.K.; Hong, C.S.; Qiao, Y.; Zhang, C. A complete survey on llm-based ai chatbots. *arXiv* **2024**, arXiv:2406.16937.
17. Weizenbaum, J. ELIZA—A computer program for the study of natural language communication between man and machine. *Commun. ACM* **1966**, *9*, 36–45. [CrossRef]
18. Wallace, R.S. *The Anatomy of ALICE*; Springer: Berlin/Heidelberg, Germany, 2009.
19. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI blog* **2019**, *1*, 9.
20. Kenton, J.D.M.W.C.; Toutanova, L.K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
21. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
22. Gao, J.; Galley, M.; Li, L. Neural approaches to conversational AI. In Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 1371–1374.
23. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6000–6010. [CrossRef]
24. Sukhbaatar, S.; Szlam, A.; Weston, J.; Fergus, R. End-to-end memory networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 2440–2448. [CrossRef]
25. Li, J.; Monroe, W.; Ritter, A.; Galley, M.; Gao, J.; Jurafsky, D. Deep reinforcement learning for dialogue generation. *arXiv* **2016**, arXiv:1606.01541. [CrossRef]

26. Izacard, G.; Grave, E. Leveraging passage retrieval with generative models for open domain question answering. *arXiv* **2020**, arXiv:2007.01282.
27. Zhang, Y.; Sun, S.; Galley, M.; Chen, Y.C.; Brockett, C.; Gao, X.; Gao, J.; Liu, J.; Dolan, W.B. DIALOGPT: Large-Scale Generative Pre-training for Conversational Response Generation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Online, 5–10 July 2020; pp. 270–278.
28. Fan, A.; Gardent, C.; Braud, C.; Bordes, A. Using Local Knowledge Graph Construction to Scale Seq2Seq Models to Multi-Document Inputs. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Hong Kong, China, 3–7 November 2019.
29. Hu, B.; Sheng, Q.; Cao, J.; Li, Y.; Wang, D. Llm-generated fake news induces truth decay in news ecosystem: A case study on neural news recommendation. In Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, Padua, Italy, 13–18 July 2025; pp. 435–445.
30. Hang, C.N.; Yu, P.D.; Tan, C.W. TrumorGPT: Graph-Based Retrieval-Augmented Large Language Model for Fact-Checking. *IEEE Trans. Artif. Intell.* **2025**, 1–15. [CrossRef]
31. Karpukhin, V.; Oguz, B.; Min, S.; Lewis, P.; Wu, L.; Edunov, S.; Chen, D.; Yih, W.t. Dense Passage Retrieval for Open-Domain Question Answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 16–20 November 2020; pp. 6769–6781.
32. Guu, K.; Lee, K.; Tung, Z.; Pasupat, P.; Chang, M. Retrieval augmented language model pre-training. In Proceedings of the International Conference on Machine Learning, PMLR, Vienna, Austria, 13–18 July 2020; pp. 3929–3938.
33. Wu, J.; Zhu, J.; Qi, Y.; Chen, J.; Xu, M.; Menolascina, F.; Jin, Y.; Grau, V. Medical Graph RAG: Evidence-based Medical Large Language Model via Graph Retrieval-Augmented Generation. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vienna, Austria, 27 July–1 August 2025; Che, W., Nabende, J., Shutova, E., Pilehvar, M.T., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2025; pp. 28443–28467. [CrossRef]
34. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
35. Reynolds, L.; McDonell, K. Prompt programming for large language models: Beyond the few-shot paradigm. In Proceedings of the Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems, Yokohama, Japan, 8–13 May 2021; pp. 1–7.
36. Shin, T.; Razeghi, Y.; Logan, R.L., IV; Wallace, E.; Singh, S. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 16–20 November 2020; pp. 4222–4235.
37. Xu, Z.; Wang, C.; Qiu, M.; Luo, F.; Xu, R.; Huang, S.; Huang, J. Making pre-trained language models end-to-end few-shot learners with contrastive prompt tuning. In Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, Singapore, 27 February–3 March 2023; pp. 438–446.
38. Miller, A.; Feng, W.; Batra, D.; Bordes, A.; Fisch, A.; Lu, J.; Parikh, D.; Weston, J. ParlAI: A Dialog Research Software Platform. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Copenhagen, Denmark, 9–11 September 2017; pp. 79–84.
39. Wei, J.; Bosma, M.; Zhao, V.; Guu, K.; Yu, A.W.; Lester, B.; Du, N.; Dai, A.M.; Le, Q.V. Finetuned Language Models are Zero-Shot Learners. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 4–8 May 2021.
40. Fregly, C.; Barth, A.; Eigenbrode, S. *Generative AI on AWS: Building Context-Aware Multimodal Reasoning Applicaions*; O'Reilly Media: Sebastopol, CA, USA, 2023.
41. Amershi, S.; Begel, A.; Bird, C.; DeLine, R.; Gall, H.; Kamar, E.; Nagappan, N.; Nushi, B.; Zimmermann, T. Software engineering for machine learning: A case study. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), Montreal, QC, Canada, 25–31 May 2019; pp. 291–300.
42. Juvekar, K.; Purwar, A. COS-Mix: Cosine Similarity and Distance Fusion for Improved Information Retrieval. *arXiv* **2024**, arXiv:2406.00638. [CrossRef]
43. Schmidt, D.C.; Spencer-Smith, J.; Fu, Q.; White, J. Cataloging Prompt Patterns to Enhance the Discipline of Prompt Engineering. 2023. Available online: https://www.dre.vanderbilt.edu/~schmidt/PDF/ADA_Europe_Position_Paper.pdf (accessed on 25 September 2023).
44. Ip, J.; Vongthongsri, K. deepeval Version 3.6.9. 2025. Available online: https://github.com/confident-ai/deepeval (accessed on 10 August 2025).
45. Topsakal, O.; Akinci, T.C. Creating large language model applications utilizing langchain: A primer on developing LLM apps fast. In Proceedings of the International Conference on Applied Engineering and Natural Sciences, Konya, Turkey, 10–12 July 2023; Volume 1, pp. 1050–1056.
46. Zaharia, M.; Chen, A.; Davidson, A.; Ghodsi, A.; Hong, S.A.; Konwinski, A.; Murching, S.; Nykodym, T.; Ogilvie, P.; Parkhe, M.; et al. Accelerating the machine learning lifecycle with MLflow. *IEEE Data Eng. Bull.* **2018**, *41*, 39–45.

47. Desai, M.; Mehta, R.G.; Rana, D.P. A Model to Identify Redundancy and Relevancy in Question-Answer Systems of Digital Scholarly Platforms. *Procedia Comput. Sci.* **2023**, *218*, 2383–2391. [CrossRef]
48. Sheng, E.; Chang, K.W.; Natarajan, P.; Peng, N. The woman worked as a babysitter: On biases in language generation. *arXiv* **2019**, arXiv:1909.01326. [CrossRef]
49. Chetnani, Y.P. Evaluating the Impact of Model Size on Toxicity and Stereotyping in Generative LLM. Ph.D. Thesis, State University of New York at Buffalo, Buffalo, NY, USA, 2023.
50. Maynez, J.; Narayan, S.; Bohnet, B.; McDonald, R. On faithfulness and factuality in abstractive summarization. *arXiv* **2020**, arXiv:2005.00661. [CrossRef]
51. Rawte, V.; Tonmoy, S.; Rajbangshi, K.; Nag, S.; Chadha, A.; Sheth, A.P.; Das, A. FACTOID: FACtual enTailment fOr hallucInation Detection. *arXiv* **2024**, arXiv:2403.19113.